

# Towards Parallelized Convolutional Dictionary Learning

Trokon Johnson<sup>1,3</sup>, Rachel LeCover<sup>2,3</sup>, Cristina Garcia Cardona<sup>3</sup>, Brendt Wohlberg<sup>3</sup>, Erik Skau<sup>3</sup>

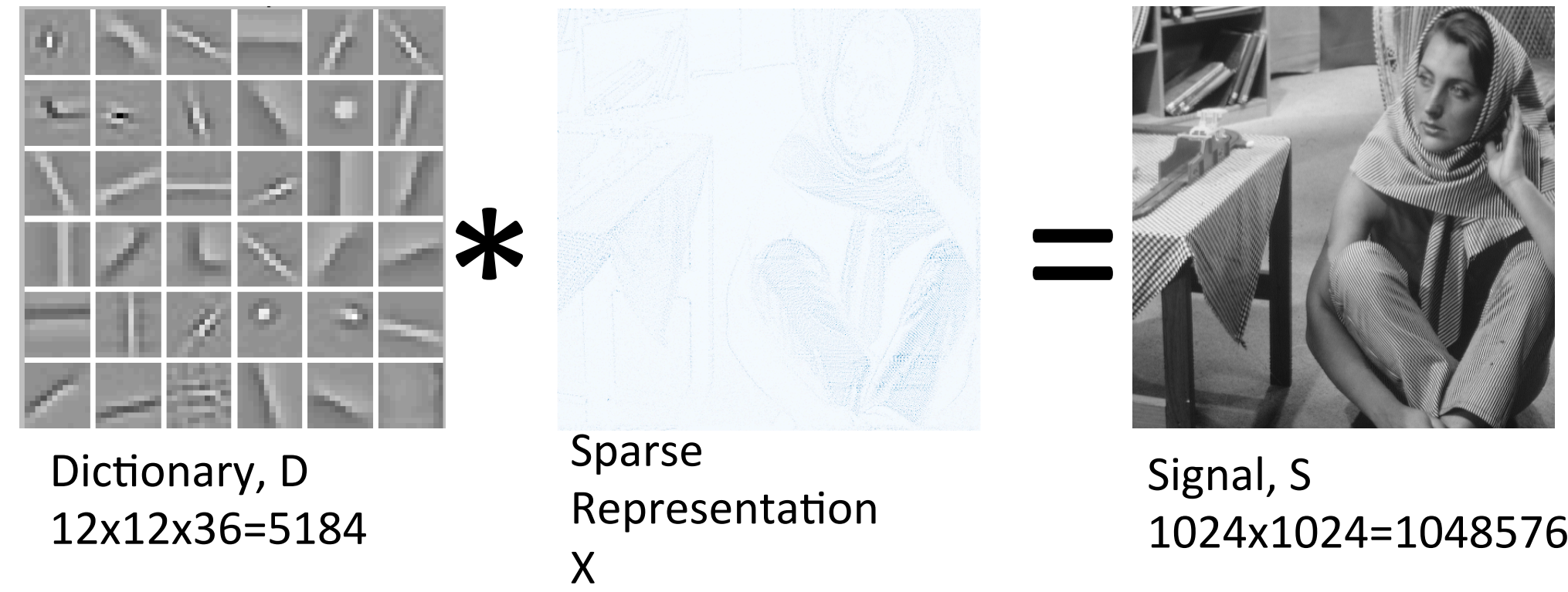
<sup>1</sup>Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida

<sup>2</sup>Robert W Smith School of Chemical and Biomolecular Engineering, Cornell University, Ithaca, New York

<sup>3</sup>Los Alamos National Lab, Los Alamos, New Mexico

## Motivation

Sparse coding and dictionary learning are two powerful techniques used to develop efficient representations of images. These representations can be used in a wide variety of applications, including image restoration problems such as denoising, and the identification of features in biological or medical images. These techniques seek to learn a dictionary, a set of patches or convolutional filters, as well as a set of coefficients that constitute the sparse representation of a signal. The original signal can then be reconstructed by convolving the dictionary and sparse representation (operation denoted by  $*$ ).



Dictionary, D  
12x12x36=5184

Sparse Representation X

Signal, S  
1024x1024=1048576

We opted to learn convolutional dictionaries, which have advantages over patch-based dictionaries as they provide translation invariant image features, which can aid the analysis of the signal structure. In this work, we present performance and scaling improvements to SPORCO (SParse Optimization Research CODE), a Python package that solves the optimization problems that arise when applying these methods.

## Background

These methods are widely used in signal and image processing applications, since they provide a compact, low dimensional representation. The sparse coding problem can be stated as:

$$\arg \min_{\{\mathbf{x}_m\}} \frac{1}{2} \left\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1$$

Where  $\{\mathbf{d}_m\}$  is a set of M dictionary filters,  $\mathbf{s}$  is the signal,  $\{\mathbf{x}_m\}$  is a set of coefficient maps, and the  $\ell_1$  norm represents a sparsity inducing penalty function. The sparse coding problem can be solved with the Alternating Direction Method of Multipliers (ADMM). As each of the k signals (images) will have its own sparse representation, we can solve all of the sparse coding problems in parallel, and then use these representations to calculate an updated dictionary.

We can state the dictionary update problem as

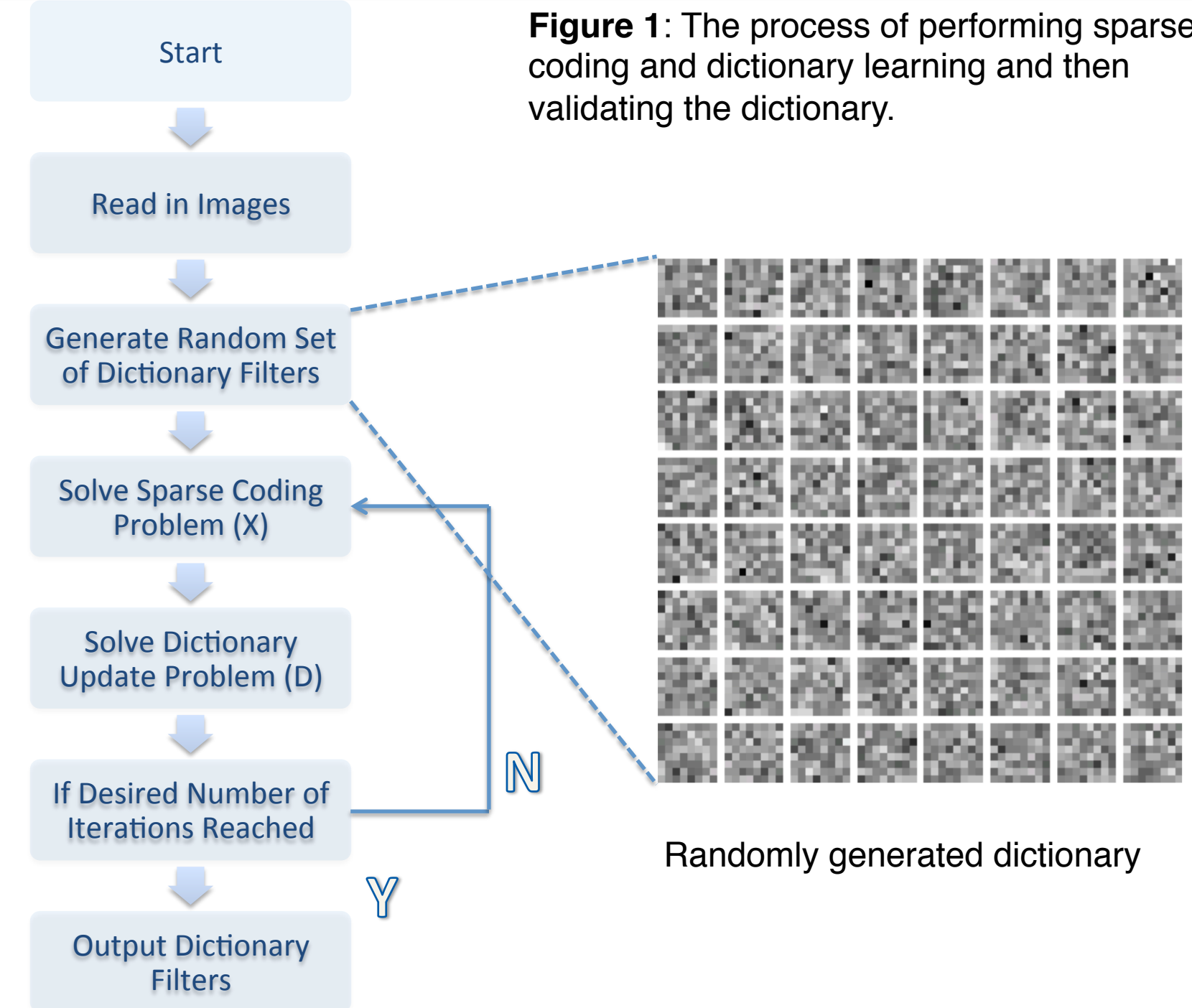
$$\arg \min_{\{\mathbf{d}_m\}} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k \right\|_2^2 \text{ such that } \|\mathbf{d}_m\|_2 = 1 \forall m$$

The dictionary filters are constrained so that their Euclidian norm must be equal to one to avoid the scaling ambiguity between  $\{\mathbf{d}_m\}$  and  $\mathbf{x}$ . Together, these two problems can be combined to form the dictionary learning problem, which is a biconvex optimization problem:

$$\arg \min_{\{\mathbf{d}_m\}, \{\mathbf{x}_{k,m}\}} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k \right\|_2^2 + \lambda \sum_k \sum_m \|\mathbf{x}_{k,m}\|_1$$

such that  $\|\mathbf{d}_m\|_2 = 1 \forall m$

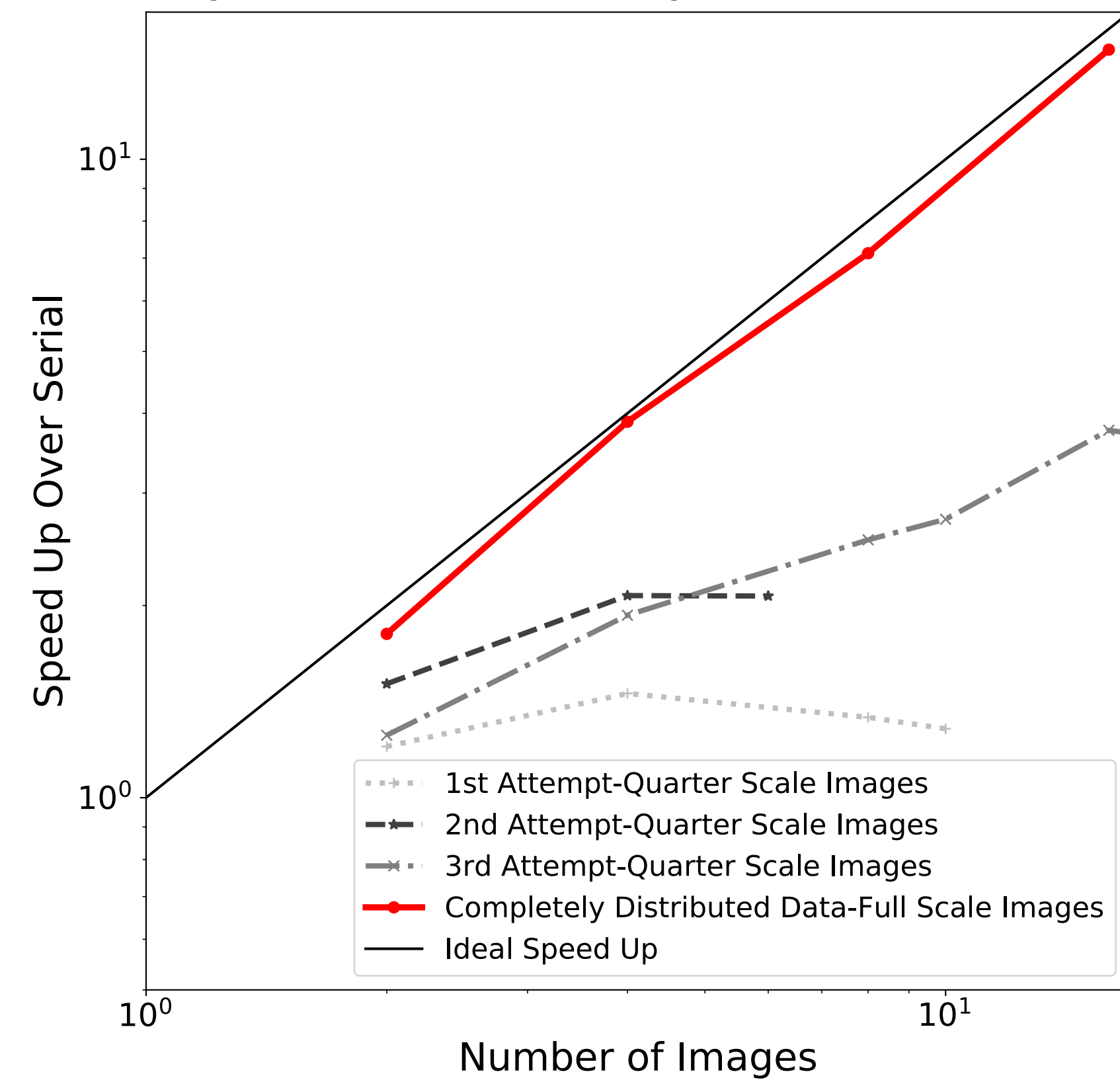
where we can alternate between solving for D and x until we reach a good enough solution or reach a target number of iterations.



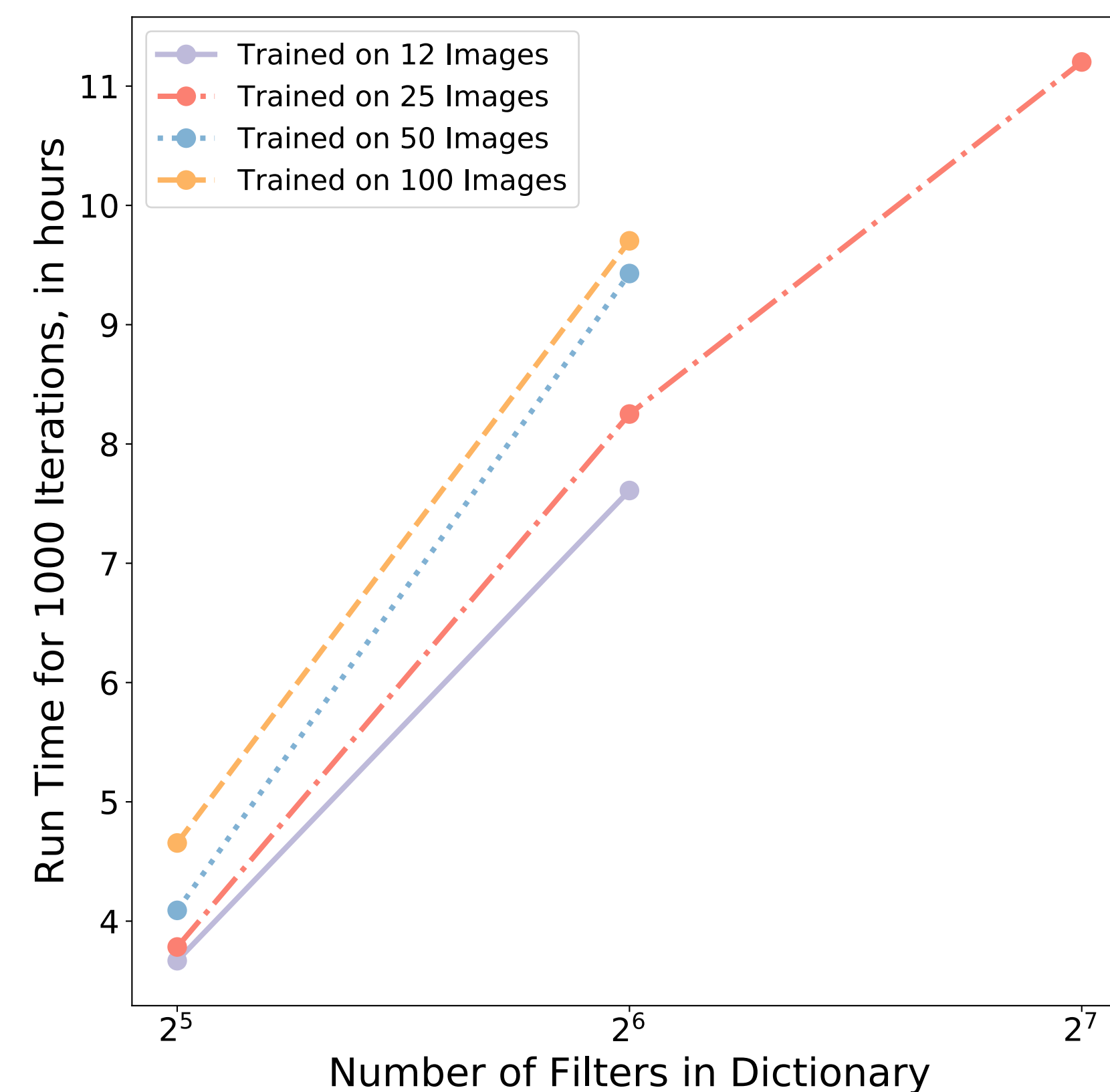
**Figure 1:** The process of performing sparse coding and dictionary learning and then validating the dictionary.

## Parallelization

ADMM consensus allows us to pose the dictionary update problem as k subproblems that can be solved independently from each other up until the consensus step, which averages the individual dictionaries created for the individual images into one overall dictionary, and therefore can be computed in parallel. We used the mpi4py package, which allows Python to use MPI functionality, to distribute the data over several nodes, with each rank holding the data from one signal.



**Figure 2:** In the final implementation, the speed up of the parallel version approached the ideal value.

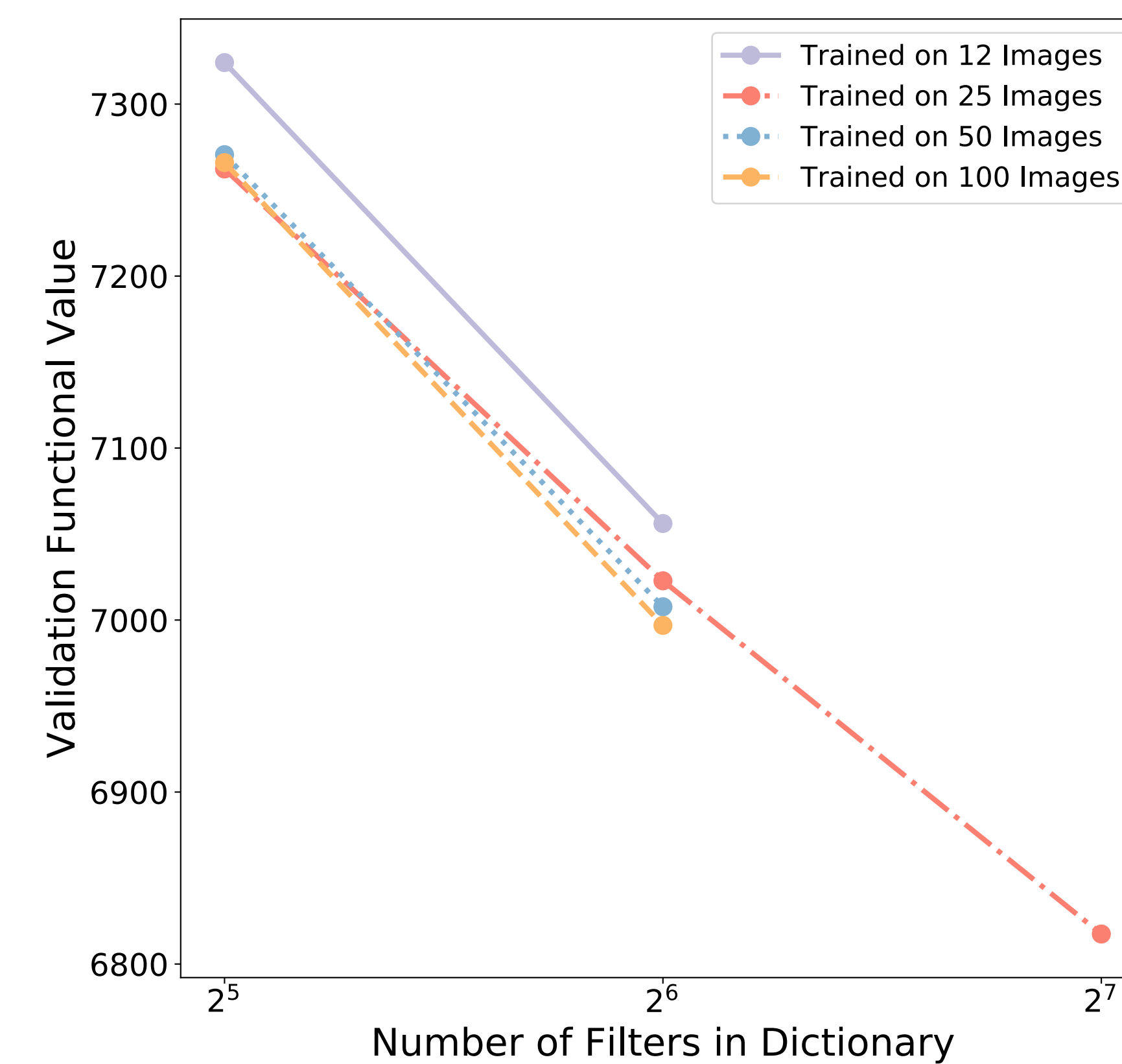


**Figure 3:** As the number of images and filters increases, so does the time to compute the dictionary.

## Results

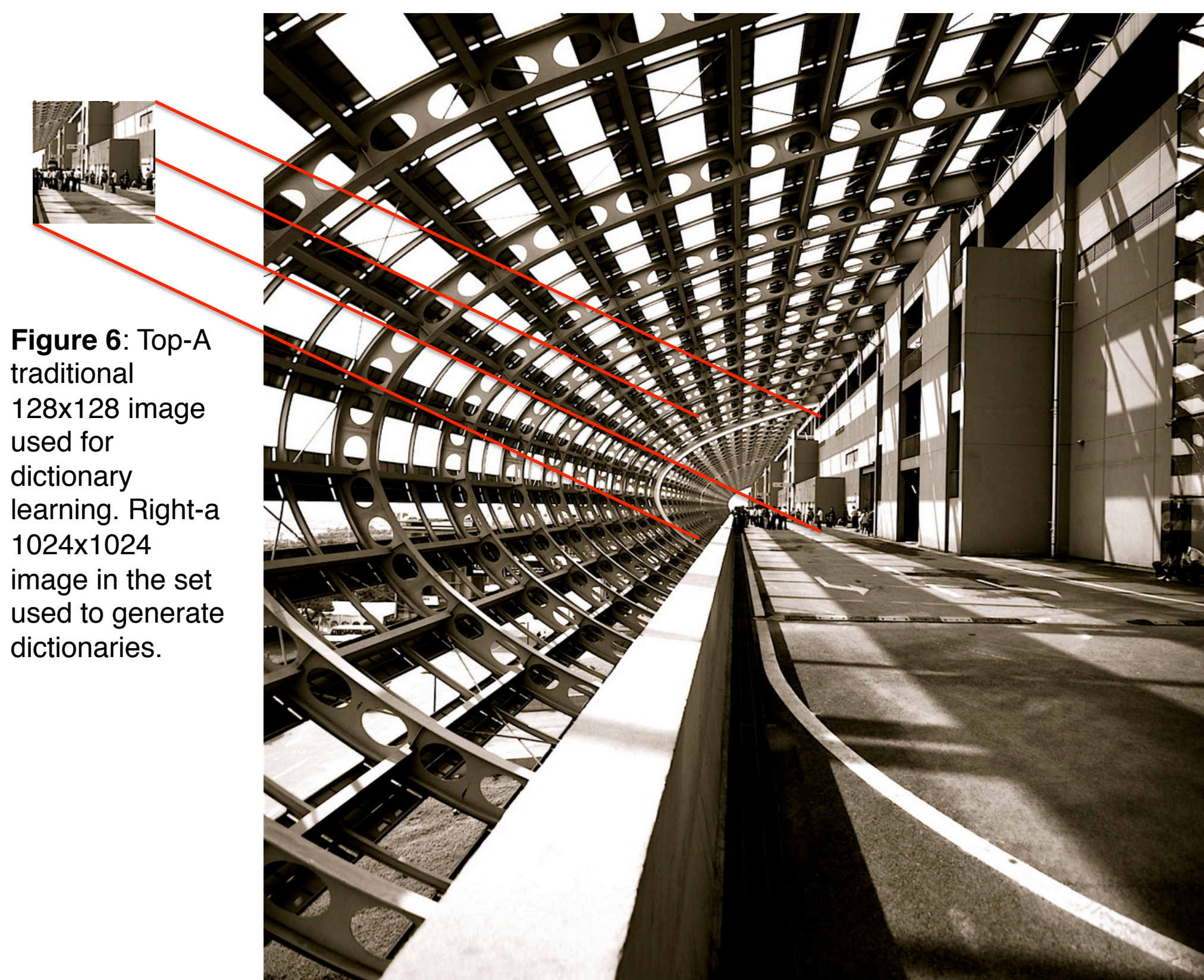


**Figure 4:** The left shows the original image, the right shows a reconstruction of the image, using a dictionary containing 32 filters created from training on 25 images and a learned sparse encoding.

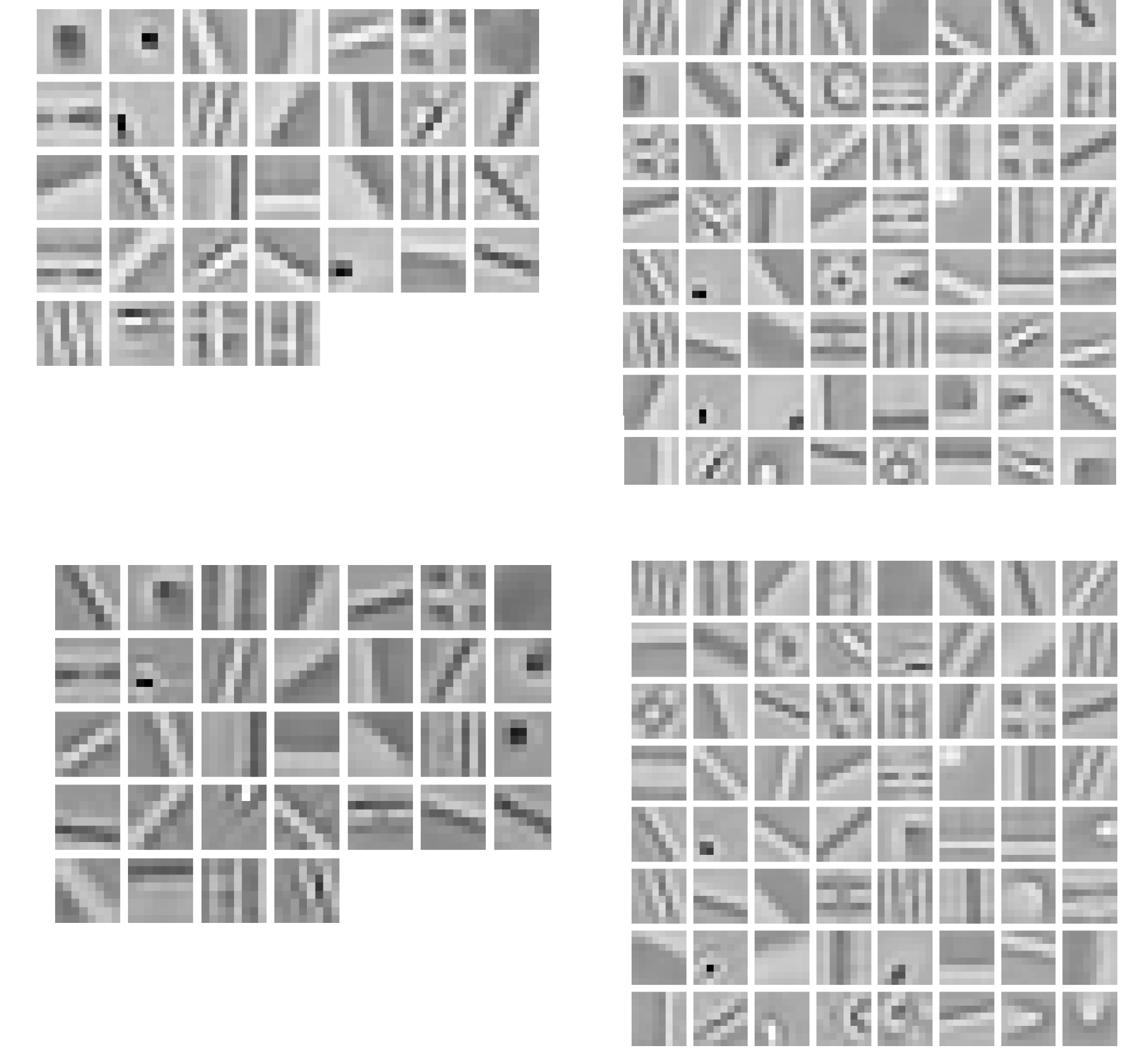


**Figure 5:** As we increase the number of images on which we train, and the number of elements in the dictionary, the performance of the dictionary increases, as shown in the lower functional value. The functional value was calculated from using the learned dictionary on 10 images not in the training set. Computations were performed with 8 ranks per node for cases of 32 and 64 filters, 3 ranks per node for 128 filters. All nodes contained at least 128 GB of RAM.

To our knowledge, this is the first experiment performed to test the efficacy of learned convolutional dictionaries based on the number of images from which they were generated as well as the number of filters they contain.



**Figure 6:** Top-A traditional 128x128 image used for dictionary learning. Right-a 1024x1024 image in the set used to generate dictionaries.



**Figure 7:** Dictionaries with 32 (left) and 64 (right) filters. The top row comes from training on a set of 12 images, the bottom a set of 25 images.

## Conclusions

By using parallelization techniques to decompose the problem into independent tasks, we developed the code to enable the largest convolutional dictionary problem ever solved to our knowledge, containing 100 large (1024x1024) images. This code will be added to the open source SPORCO package to be freely used. We observed that as the number of images and filters used increased, so did the performance of the dictionary when performing sparse coding.

## References

- Wohlberg, Brendt. "Efficient algorithms for convolutional sparse representations." *IEEE Transactions on Image Processing* 25.1 (2016): 301-315.
- Šorel, Michal, and Filip Šroubek. "Fast convolutional sparse coding using matrix inversion lemma." *Digital Signal Processing* 55 (2016): 44-51.
- SPORCO - <http://bwohlberg.github.io/sporco/>

## Acknowledgements

This work was carried out under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory supported by contract #DE-AC52-06NA25396 as well as by the U.S. Department of Energy through the LANL/LDRD Program and by UC Lab Fees Research grant 12-LR-236660.

TJ received additional support from DOE PSAAP 2.

Computations were performed using the Darwin Computational Cluster and HPC resources at Los Alamos National Laboratory.